IN THE SPECIFICATION

Please amend the specification as follows. Paragraphs that are being amended are listed in their entirety; changes are indicated in the left margin with a vertical change bar. Deletions are marked by ~~strikethrough~~; insertions are underlined.

Please amend the paragraph beginning on page 5, line 20, through page 6, line 6, as follows:

Combinational circuits are generally thought of as acyclic structures and sequential circuits as cyclic structures. A collection of logic gates connected in an acyclic, or loop-free, topology is combinational because, regardless of the initial values of its inputs, once the inputs are fixed the signals propagate to the outputs. There is a direct correspondence between the electrical behavior of the circuit and the abstract notion of the boolean functions that it implements. The behavior of a circuit with feedback, or cycles, is generally more complicated. Such a circuit may exhibit sequential behavior, such as an R-S Latch, or ~~its~~ it may be unstable, such as an oscillator.

Please amend the paragraph beginning on page 14, line 12, through page 15, line 15, as follows:

In the following description, standard notation is used: addition (+) denotes disjunction (OR), multiplication denotes conjunction (AND), and an overline ($\overline{x}$) denotes negation (NOT). A model is used that is at a level of abstraction applicable in the technology-independent phase of logic synthesis. A network is constructed that computes boolean target functions $g_i(x_1, \ldots, x_m)$, $1 \leq i \leq m$ of boolean input variables $x_1, \ldots, x_m$. Internally, the network is specified as a collection of nodes and associated with each node is a node function f_1 and an internal variable y_i, $1 \leq i \leq n$. The node ~~function~~ functions depend on internal variables and input variables. A directed edge

is drawn from node i to node j if and only if the node function f_j associated with node j depends on the internal variable y_i associated with node i. A subset of the nodes are designated as output nodes. For output nodes, the ~~targets~~ target functions are the requisite output functions. In the following discussion, for the sake of readability, the same symbol for the node function, the target function and the associated internal variable are used in equations. For example, when a symbol f_i is used on the left-hand side of an equation it refers to a function, and when used on the right-hand side it refers to the corresponding internal variable.

A network is combinational if and only ~~of~~ if it computes unique boolean output values for each boolean input ~~assignments~~ assignment. If there are "don't care" conditions on the inputs, then it is sufficient if the network computes unique boolean values for input assignments in the "care" set. This computation must hold regardless of the initial state and independently of all timing assumptions. As noted earlier, a cost of a network can be estimated by a measure of the sum of the literals in the factored form of the node expressions of the network. See R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis"; Proceedings of the IEEE, Vol. 78, No. 2, pp 264-300, 1990.

Please amend the paragraph beginning on page 19, line 14, through page 20, line 14, as follows:

As noted, in existing methodologies, a total ordering is enforced among the functions in the substitution phase in order to ensure that no cycles occur. This choice can influence the cost of the solution. For instance, with the ordering shown in Figure 10, substitution yields:

$$f_1 = g_1(a,b,c,) = a(b\bar{c} + \bar{b}c)$$
$$f_2 = g_2(a,b,f_1) = b\bar{f}_1 + a\bar{b}$$
$$f_3 = g_3(a,b,c,f_1) = a\bar{f}_1 + b\bar{c}$$

with a cost of 13, whereas the ordering shown in Figure 11 yields:

$$f_1 = \bar{b}cf_2 + b\bar{f}_2$$
$$f_2 = b(c + \bar{a}) + a\bar{b}$$
$$f_3 = a\bar{f}_1 + b\bar{c}$$

with cost 14. As ~~noted~~ noted, an ordering is limiting because functions near the top cannot be expressed in terms of very many others. As illustrated, removal of this restriction can lower the cost. For example, if we allow every function to be substituted into every other, the network can be expressed as:

$$f_1 = \bar{f}_2 b + \bar{f}_3 a$$
$$f_2 = \bar{f}_1 b + a\bar{b}$$
$$f_3 = \bar{f}_1 a + b\bar{c}$$

with a cost of only 12. This network is cyclic, and not combinational. This may be verified according to well known procedures. For example, note that when $a = 1$ and $b = 1$, then $f_1 = \bar{f}_2 + \bar{f}_3, f_2 = f_3 = \bar{f}_1$. Note that if the order of substitution is restricted to that shown in Figure 12, the network can be expressed as:

$$f_1 = a(\bar{f}_3 + b\bar{c})$$
$$f_2 = b\bar{f}_1 + a\bar{b}$$
$$f_3 = \bar{c}f_2 + ab$$

the network is combinational ~~an~~ and has a cost of 12.

Please amend the paragraph on page 23, line 4 through page 24, line 8, as follows:

The "Break-Down" Approach:

With this approach, the search is performed outside the space of combinational solutions. A branch terminates when it hits a combinational solution. The search begins with a maximally connected network as shown in Figure 14. For each node f_i, a node expression $g_i^{(1)}$ is generated based on the complete substitutional set, i.e., all other nodes in the network. The dependency of a node function on another node is referred to as an "edge". This initial branch has the densest set of edges, and its cost provides a lower bound on the cost of the solution. As edges are excluded in the branch-and-bound process, the cost of the network remains unchanged or increases. Again, since the substitution step is based on heuristics, this may not be strictly true. In the "break-down" approach the following steps are followed:

1. Analyze current branch for combinationality. If it is combinational, add it to the solution list. If it is not, select a set of edges to exclude based on the analysis,.
2. For each edge in the set, create a new branch. Create a node expression, excluding the incident node from the substitutional set. If the cost of the new branch ~~equal~~ equals or exceeds that of a solution already found, kill the branch.
3. Mark the current branch as "explored."
4. Set the current branch to be the lowest-cost unexplored branch.
5. Repeat steps 1 - 4 until the cost goal is met.

A sketch of the "break-down" approach is illustrated in Figure 14. Figure 14 does not illustrate a complete trace of the search, it only illustrates a trajectory to the solution shown. As shown in Figure 14, the "break-down" approach yields a cyclic combinational solution with a cost of 13. This network can be mapped, for example, to a circuit with 9 fan-in two gates as shown in Figure 24.

Please amend the paragraph on page 24, line 18 through page 25, line 2, as follows:

Figure 15 is a flow chart illustrating the "break-down" approach illustrated in Figure 15. Flow begins in block 1502 where a densely interconnected network is created. Flow then continues to block 1504 where the network is analyzed to establish the lower cost boundary of the network. Flow then continues to block 1506 ~~were~~ where a set of edges is excluded from an unexplored branch of the network.

Please amend the paragraph on page 25, line 7 through page 26, line 11, as follows:

Returning to block 1508, if the network is determined to be combinational then flow continues to block 1512. In block 1512 the cost of the network is determined and evaluated to determine if a cost goal has been met. If the cost goal has not been met flow continues to block 1514 where the cost of the current network is compared to the cost of solutions that have already been found. If the cost of the current network is greater than, or equal to, the cost of a solution that has already been found flow continues to block 1510 and the branch is terminated and marked as explored. If, in block 1514 it is determined that the cost of the current network is less than any solutions that have already been found, then flow continues to block 1516. In block 1516 the current branch is ~~set t~~ set to the lowest cost unexplored branch. Flow then continues to 1506.

Returning to block 1512, if it is determined that the cost goal has been met, then flow continues to block 1518 and flow terminates.

The ~~"Build-Up "~~ "Build-Up" Approach:

With this approach, a search is performed inside the space of combinational solutions. A branch terminates when it hits a non-combinational solution. The search begins with an empty edge set, that is the target functions. Edges are added as the

substitutional sets of nodes are augmented. As edges are included, the cost of the network decreases. The process is:

1. Analyze current branch for combinationality. If it is not combinational discard it. If it is combinational, select a set of edges to include based on the ~~analysis,~~ analysis.
2. For each edge in the set, create a new branch. Create a new node expression, including the incident node from the substitution set.
3. Mark the current branch as "explored".
4. Set the current branch to be the lowest-cost unexplored branch.
5. Repeat steps 1 - 4 until the cost goal is met.

Please amend the paragraph on page 26, line 16 through page 27, line 4, as follows:

With the "build-up" approach ~~braches~~ branches cannot be pruned through a lower-bound analysis. However, exploring within the space of combinational solutions ensures that incrementally better solutions are found as the computation proceeds. As an alternative existing acyclic solutions may be used as a starting point. Adding edges reduces costs but potentially introduces cycles.

In general, the "break-down" approach performs best on dense examples, whereas the "build-up" approach performs better on sparse examples. In addition, a hybrid "build-up"/"break-down" approach ~~is~~ may be feasible where features of the two approaches are combined to arrive at a solution.

Please amend the paragraph on page 28, lines 14-20, as follows:

A symbolic framework is formulated for analysis that obviates the need for ternary-valued simulation. The problem is approached with a "divide-and-conquer" technique: progressively smaller components of the network are analyzed for

combinationality. It is noted that if a network's dependency graph can be divided into several distinct strongly connected components, then the analysis may be performed separately on each component. For simplicity, it is ~~assume~~ assumed that each node in the network is an output node.

Please amend the paragraph on page 42, lines 4-13, as follows:

As illustrated, a paradigm shift in combinational circuit design is appropriate: combinational logic should no longer be thought of as acyclic in theory or in practice, since nearly all combinational circuits are best designed with cycles. A general methodology has been formulated for the synthesis of cyclic combinational circuits, and incorporated into a logic synthesis environment. Search ~~process~~ processes have been described that, while heuristic, can effectively tackle circuits of sizes that are of practical importance. The implementation of more sophisticated search processes, such as stochastic search, and parallelization are obvious evolutions of the project. Furthermore, the techniques described can be extended to the technology mapping phase of logic synthesis.